

PRVPATENT- OCH REGISTRERINGSVERKET
Patentavdelningen

REC'D PCT/PTO

02 MAR 2003

3

**Intyg
Certificate**

Härmed intygas att bifogade kopior överensstämmer med de handlingar som ursprungligen ingivits till Patent- och registreringsverket i nedannämnda ansökan.

This is to certify that the annexed is a true copy of the documents as originally filed with the Patent- and Registration Office in connection with the following patent application.



(71) Sökande Sixsteps AB Ideon Science Park, Lund SE
Applicant (s)

(21) Patentansökningsnummer 0202593-0
Patent application number

(86) Ingivningsdatum 2002-09-03
Date of filing

REC'D 16 SEP 2003

WIPO PCT

Stockholm, 2003-09-05

För Patent- och registreringsverket
For the Patent- and Registration Office

Lisa Junegren

Avgift
Fee

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

AWAPATENT AB

Kontor/Handläggare
Malmö/BAN

SixSteps AB

Ansökningsnr

Vår referens
2010605

1

A COMPUTER PROGRAM PRODUCT AND ASSOCIATED METHODS FOR
SEARCHING A DATABASE OF OBJECTS, CONNECTING OBJECTS IN
SUCH A DATABASE, AND EXPORTING DATA FROM AT LEAST ONE
ARBITRARY DATABASE

Technical Field

The present invention generally relates to the field
of finding, connecting and browsing any number of in-
5 dependent information objects that may or may not refer
to each other by means of any type of attribute or value,
as long as every information object consists of at least
one data element attribute and one data element value.
More particularly, the invention relates to a computer
10 program product and associated methods of searching a
database of objects, connecting objects in such a data-
base, and exporting data from at least one arbitrary
database.

15 Background Of The Invention

Most knowledge producing organizations have a
project oriented value chain that utilizes the expertise
of different line functions to contribute to the end
result. Since the expert line functions are using
20 different information systems to efficiently carry out
their tasks, there is a need for an information system
that can integrate and preserve the obtained digital
information in a flexible and easy way. The need is both
to preserve data for extended time periods, and to be
25 able to easily re-use them in new constellations. This
problem is a reality in most cumulative knowledge
building activities ranging from the academic world to
commercial pharmaceutical research. However, it is also
an identified problem in modern healthcare, where vital
30 patient information may come from many different labora-

tory or hospital systems, but still need to be stored, integrated and re-used for a lifetime.

Current technologies for addressing these problems are based on relational databases or object oriented
5 databases. These technologies have proved very efficient in creating systems for data transaction and reporting. However, they are not designed for long time storage and an open integration between different systems, and they are not easily altered to accommodate new types of data.
10 Although object-oriented databases handle objects, they are not suitable for integration of different types of objects in a distributed environment. The reason for this is that in an object-oriented database fixed classes have to be defined, and all data has to fit into the structure
15 of those classes. Searching is difficult since the user needs to know in which class and in which attribute data should be searched for. Relations between objects are "fixed links" which must be known a priori.

Hence, neither of these technologies is suitable for
20 obtaining the ease of integration between distributed databases that is the demand. There is clearly a need to improve the situation by suggesting technologies that are more focused on keeping and integrating individual knowledge elements into accumulated knowledge.

25

Summary Of The Invention

Accordingly, an objective of the present invention is to mitigate the above-mentioned shortcomings and to provide a new method for retrieval and integration of any
30 number of independent information objects, that may be stored in a standardized future-proof text format and that may refer to each other by means of any type of attribute or value.

This objective is achieved by a computer program
35 product and methods having the features recited in the attached independent claims. Preferred embodiments of the invention are recited in the dependent claims.

Thus, a first aspect of the invention is a computer program product having program code adapted to provide, upon execution, a database of objects and a database engine for managing said objects, where:

5 each object comprises metadata for describing the object and for defining a hierarchical structure of branches which constitutes said object and which includes relations to other objects,

wherein an individual branch has:

10 a metadata type which is selected from a pre-determined and limited set of n different metadata types;

a metadata value; and

15 an arbitrary number of other branches connected thereto, as children of said individual branch, said arbitrary number including zero branches, and

wherein each metadata type represents one respective level in said hierarchical structure.

A second aspect of the invention is a method of
20 searching a database of objects as provided by the computer program product according to the first aspect. The method involves the steps of

25 specifying, through a man-machine interface of a computer, a search query in a declarative language in accordance with said hierarchical structure of objects in said database;

submitting said search query to said database engine through said computer;

30 receiving a result of said search query at said computer; and

presenting said result through said man-machine interface.

35 A third aspect of the invention is a method of exporting data from at least one arbitrary database. The method involves the steps of

identifying a first plurality of relations between different pieces of data in said at least one database;

defining a second plurality of objects, each of said objects comprising metadata which represents individual ones of said first plurality of relations and said different pieces of data, said metadata being of n
5 different metadata types, n being a predetermined integer;

for each object, forming an n-level hierarchical information structure, where each metadata type is represented at a respective unique level; and
10 storing the information structures thus formed for said second plurality of objects.

A fourth aspect of the invention is a method of connecting objects in a database of objects as provided by the computer program product according to the first
15 aspect. The method involves the steps, for an assumed individual object, of:

for all branches in the assumed object that are of a first predetermined metadata type, said first predetermined metadata type allowing a relation to be defined
20 from the assumed object to another object:

forming a query based on the metadata of the branch,
searching the database with the query, and
collecting, as a result of the searching step;
25 all objects, if any, that the assumed object has a relation to; and

for all branches in the assumed object that are of a second predetermined metadata type, other than said first predetermined metadata type, said second predetermined
30 metadata type allowing a reverse relation to be defined from another object to the assumed object:

forming a query based on the metadata of the branch,
searching the database with the query, and
35 collecting, as a result of the searching step, all objects, if any, that have a relation to the assumed object.

According to the present invention any type of tabular or text database can be transformed into one or many specific generic information structures expressed in a future-proof text format like e.g. xml. These

5 information structures can be contained in one or many flat files that may be stored on a magnetic disk, an optical disk, a magneto-optical disk or an electronic semiconductor memory for extended times.

The text-based information structures can be
10 automatically indexed and put into a strictly hierarchical n-field structure that may reside on secondary memory, but still be easily and rapidly queried by a simple query syntax. This client-server architecture also makes it possible to incrementally update the data-
15 base online.

The minimal n-field structure can be indexed by means of reverse indices for all fields, which makes it possible to perform exhaustive searches of the database from a single field without any prior knowledge about the
20 database structure or content. The search can be specified as regards attribute, type and value and allows for full Boolean logic as well as "starts with" or "fuzzy logic". These search facilities, together with the strict hierarchical nature of the database structure, makes it
25 possible to easily find any specific data or group of data in the database.

Preferably, the hierarchical n-field structure has 5-7 fields, i.e. $5 \leq n \leq 7$. Even more preferably, $n = 6$.

30 After finding the desired information elements, their relations with each other may appear as named links from the respective elements. By selecting such a link all objects using this link will be selected. It is in this way easy to navigate around in the network of
35 relations that connect the different related information objects. Any object can at any time be chosen as the "root" or "perspective" from which all the other objects

are seen and can be reached. An important feature in this context is the usage of reversed link technology. This makes it possible to show a relation from an object A to another object B, although the actual relation is only
5 given in object B.

In contrast to prior art database technologies where relations between objects are "fixed links", such relations may be found on the fly according to the the present invention.

10 Besides the functionality of finding and navigating in the database, the new technology also permits the use of attributes on objects, links, and metadata. This opens up the possibility to introduce an explicit time dimension in the database. Hence, objects and relations
15 can be initiated or updated without losing the audit trail by simply setting an end date on previous values and/or relations. In subsequent searches and navigations a filter can be used for obtaining only the values that are or were valid at any specific point in time. This
20 greatly facilitates the monitoring of sequences of time-based events that is necessary for tracking audit trails, organizational changes, validity of standard operating procedures etc.

Another functionality unique to the proposed technol-
25 ogy is the use of predefined associations. Associations are described by a hierarchy of directed paths between object types and are automatically processed in the search algorithm. Any number of associations can be defined and applied. This feature can be used within any
30 given local database or group of databases and will improve the speed of search within that local database or group of databases without hampering the total flexibility of the distributed nature of the invention. It is in this way possible to rapidly integrate and recreate large
35 databases.

Generally, all terms used in the claims are to be interpreted according to their ordinary meaning in the

technical field, unless explicitly defined otherwise herein. All references to "a/an/the [element, means, component, member, unit, step etc.]" are to be interpreted openly as referring to at least one instance of said element, means, component, member, unit, step etc. The steps of the methods described herein do not have to be performed in the exact order disclosed, unless explicitly specified.

10 Brief Description Of The Drawings

One embodiment of the present invention will now be described by way of examples, reference being made to the accompanying drawings, in which

FIG. 1 is a diagram providing a general overview of the search algorithm according to the invention;

FIG. 2 is a diagram that in detail describes the step of creating a hierarchy of row searches and joins;

FIG. 3 is a diagram that in detail describes the step of creating a hierarchy of searches and joins with attribute constraints;

FIG. 4 is a diagram that in detail describes the step of performing the row searches;

FIG. 5 is a diagram that in detail describes the step of performing successive joins in the hierarchy of row searches and joins;

FIG. 6 is a diagram that in detail describes the step of performing successive joins in the hierarchy of row searches and joins with attribute constraints;

FIG. 7 is a diagram providing a general overview of the search algorithm with association maps or inheritance of metadata according to the invention;

FIG. 8 is a diagram providing a general overview of the system model of the present invention;

FIG. 9 is a diagram providing a general overview of a distributed architecture in which the present invention may be applied; and

FIG. 10 is a diagram that illustrates an algorithm for finding other objects by using relations from and reverse relations to an individual object.

5 Detailed Description Of The Invention

Objects

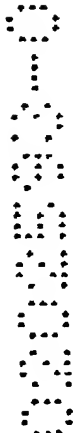
The fundamental unit of processing in the present invention is an object. An object has metadata connected
10 to it. Metadata is data describing the object. An object consists of several branches connected to each other in a hierarchical or tree-like fashion. A branch can have any number of other branches connected to it, also called the children of the branch. A branch is of a certain metadata
15 type: o (object), r (relation), k (key), a (attribute), t (type) or v (value). A branch also has a metadata value.

Any model in a relational database can be broken down into a system of the objects processed by the present invention, where one or many objects will represent
20 the rows and columns in the tables of the relational database. The relations or references between different data in a database can be described as metadata in objects and even named. In most cases the database model can be clarified by naming the relations.

25

Object structure

The structure of the object is a strict hierarchy of the six different metadata types given above. Since the metadata types are hierarchically arranged, where every
30 metadata type is represented at a certain level, they are also referred to as levels in the text below. These metadata types are the only ones allowed. However any number of branches of the same metadata type is allowed and a branch can have any number of children. Any number
35 of intermediate levels can be skipped but has to be in the correct order. It is a very flexible format with only two restrictions: 1) the root branch must always be o,



and there can only be one branch of that metadata type.

2) The tags, that is the metadata types, must always lie in the described order.

Since the objects consist of a strict hierarchy, it is very well expressed in a declarative language such as xml, which is based on the ISO standard SGML. See Table 1 below.

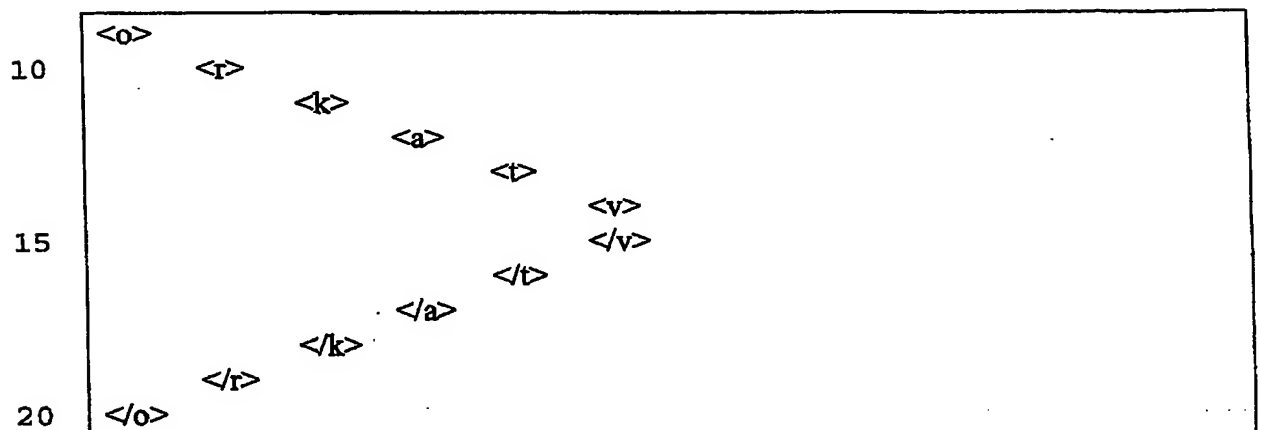


Table 1. Object structure.

Other declarative languages that can be used to express the object hierarchy include SGML dialects, hierarchially connected objects in a programming language, a graphically modelled hierarchy, etc.

The object structure can be described by a context-free grammar defined in the Chomsky hierarchy of formal languages, as seen in Table 2:

Object	→	<o> Relations </o>
Relations	→	Keys <r> String Keys </r> Relations
Keys	→	Attributes <k> String Attributes </k> Keys
Attributes	→	Types <a> String Types Attributes
Types	→	Values <t> String Values </t> Types
Values	→	s <v> String </v> Values
String	→	s Character String
Character	→	\u0001 ... \uFFFF

Table 2. Context-free grammar of the object structure

All metadata values in the examples are assumed to be expressed in Unicode strings. A simple example of an object described in xml is given in Table 3:

```
<o>Adam
  <r>wife
    <a>person
      <t>person id
        <v>500101-2221
        </v>
      </t>
    </a>
  </r>
  <a>person
    <t>person id
      <v>480101-1111
      </v>
    </t>
    <t>name
      <v>Adam
      </v>
    </t>
  </a>
</o>
```

5 Table 3. Simple object example.

By expressing the objects in xml, all the contents of a database can be moved to a future-proof format that can be stored on e.g. CD-disks, while still being actively queried and used by the present invention.

By using the objects described, all data from any database can be transformed into a simple six-field structure. In addition the type of content of the different fields is always the same. This makes it possible to search for any specific content, regardless of the initial complexity in the same manner. All regular database entries can be put in the v-field, and all relations in the r-field etc. Table 4 below gives a simple example of a transformation from the object in Table 3 to a branch table, which is shown in Table 5.

11

```

<o>Adam                                     //0
  <r>wife                                   //0
    <a>person                               //0
      <t>person id                           //0
        <v>500101-2221                       //0
        </v>
      </t>
    </a>
  </r>
  <a>person                                 //1
    <t>person id                             //1
      <v>480101-1111                         //1
      </v>
    </t>
    <t>name                                 //2
      <v>Adam                               //2
      </v>
    </t>
  </a>
</o>

```

Table 4. Simple example used for illustrating the transformation from an object to the branch table. The comments give the unique branch identifier.

O	r	k	a	t	v
0 [Adam]	0 [wife]	-	0 [person]	0 [person id]	0 [500101-2221]
0 [Adam]	-	-	1 [person]	1 [person id]	1 [480101-1111]
0 [Adam]	-	-	1 [person]	2 [name]	2 [Adam]

Table 5. The branch table created from the data in Table 4.

5

Internal format of metadata

In the present invention all objects are stored internally in the same structure as described above. A unique branch identifier represents each branch. Since all branches have a metadata type, it needs only to be unique within the metadata type. Each branch identifier has an associated value, which is the metadata value for that branch. All unique values are in the same way represented by a unique value identifier. More than one branch identifier can have the same value.

Internal structure format

All objects, including all the metadata, are also internally stored in a structure, called the *branch table*. The branch table has the same columns as the different metadata types. In the cells of the table, branch identifiers are stored. Objects are transformed into the branch table by letting every *end branch* in the objects result in a row in the table. An end branch is a branch that has no children. Each row contains all the branches that precedes the end branch as its parents, and the columns in the row contain the branch identifiers for the branches (see Table 4 and 5).

Indexing

In order to be able to quickly search for metadata values in the objects, each column in the branch table is indexed. The kind of indices that is applied to the columns can be any kind of indices used in databases, which allows searches like "begins with" or "fuzzy". Keys in the indices are the values of the branches and results of searches are the rows in the branch table. It might be the entire rows or just the row numbers. In the latter case the actual rows can be fetched later from where it is stored.

Search Algorithm

When a search is performed, only the table, its indices and the values of the branches are needed. A search query has the same structure as an object except that the root branch is not an object but rather a connecting "query" branch that connects all the separate search conditions in the query (see example query described in xml in Table 6 below). The search is a search for parts of an object. An object is considered found if it includes all the searched parts.

```

5      <query>
      <a>person
      <t>person id
      <v>480101-1111
      </v>
      </t>
      <t>name
      <v>Adam
      </v>
10     </t>
      </a>
      <r>wife
      </r>
15    </query>

```

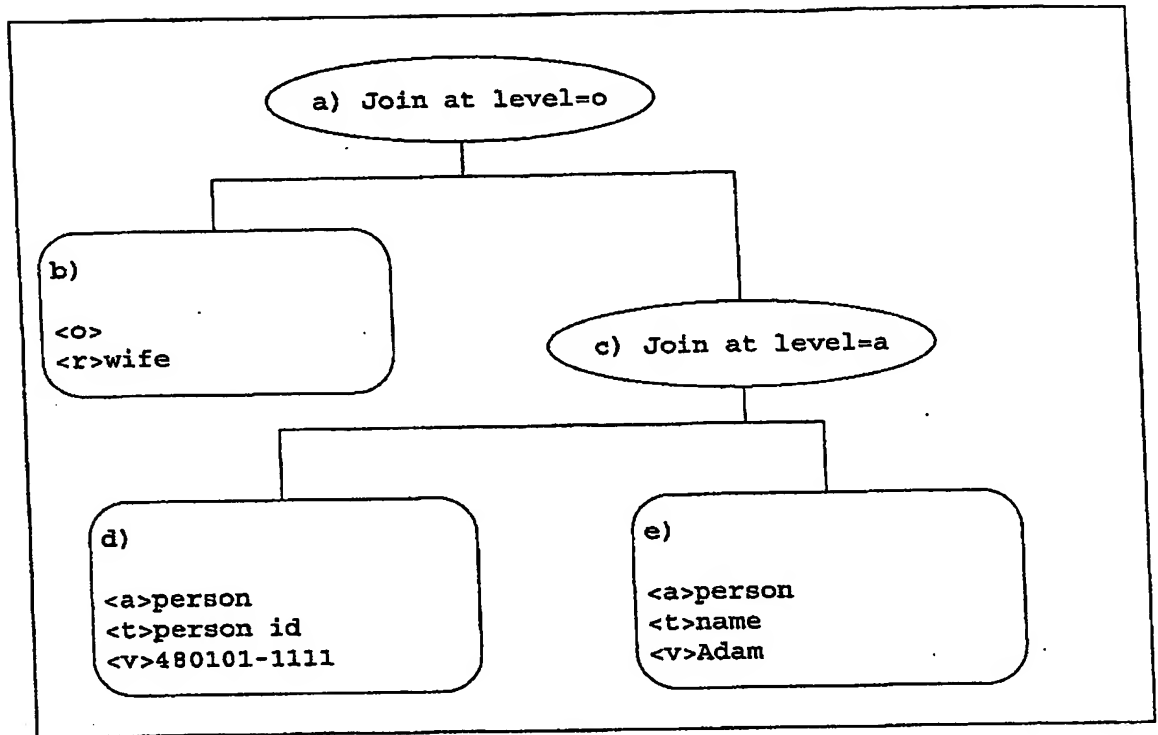
Table 6. Example of query in xml format.

A search is performed in a number of steps. An overview of the search algorithm is given in FIG. 1. A given query (step 101) is transformed into a hierarchy of so called row searches and joins (step 102). The row searches and joins are called *search units* and a hierarchy of search units is called a *search hierarchy*. In a next step all the row searches are done, which result in a number of rows in the branch table (step 103). These results are then joined successively until the top level of the search hierarchy is reached (step 104). The row searches and joins will be described in more detail in the following text. The last step of the search algorithm is to calculate the resulting objects of the search by getting all the unique objects from the resulting rows from the search units.

Given a query the present invention creates a search hierarchy (step 102). This process is described in detail in FIG. 2. The query has a number of branches. All the branches that describes a separate condition for the search and that are connected directly to the "query branch", are joined at metadata type o. A top join for that purpose is created (step 201). The query is traversed top-down and a branch is always processed before its children (step 202). For all the branches in the query the algorithm decides what to do depending on

how many children the branch has. If the branch has no children (step 203) it is an end branch and a row search is created (step 204). The metadata values in the query on all branches from the root down to the end branch, becomes the conditions on metadata values in the row search. A condition may have complex values such as "or", "begins with" or "fuzzy". A row search is a number of conditions on metadata and finds all rows in the branch table that comply with all the conditions. If the number of children of a branch in the query is more than one (step 205), a join is created (step 206). The join is given the metadata type of the query branch. All other search units that will become its children in the search hierarchy are joined at that metadata type when the search units are performed. When a search unit has been created it is added as a child to the preceding join (step 207). The preceding join is the join created for the previous parent branch with more than one child, or the top join, if no such join has been created. If the number of children of a branch in the query is exactly one, nothing is done and the metadata value of the branch will be a condition in all end branches that lies under the branch. Hence a search hierarchy is very similar in structure to the query, with the exception that all branches with exact one child do not result in search units. Given the example query in Table 6 the search hierarchy in Table 7 is created.





5 Table 7. Resulting search hierarchy from the query described in Table 6

When the search hierarchy has been created, all row searches will be performed (step 102). A row search is a search for rows in the branch table that comply with all conditions in the row search. This can be done in many ways depending on whether data is kept in primary or secondary memory. One possible method is described in detail in FIG. 4. All the row searches are done and result in a set of rows in the branch table (step 401).

10 It might be just the row numbers but it might also be the entire rows and even the values of the branch identifiers in the rows, all depending on how much primary memory is available, the required speed of a search etc. All row searches have a number of conditions on metadata values.

15 Each metadata type has a condition that contains any number of allowed values for that metadata type, and may be complex values such as "or", "begins with" or "fuzzy".

20 A condition example is seen in Table 8:

<v>"Ohlson" OR "Ols"*

Table 8. Example condition. Allowed values for metadata type v are "Ohlson" or all values that begins with "Ols".

5 For all metadata types (step 402) the algorithm gets how many rows the allowed values for that metadata type would result in (step 403). This is done quickly since the indices of the present invention allow estimation of this in constant time even for more complex value
10 conditions such as "begins with". When it is found out what metadata type will give the least number of rows, the search for the allowed values of that metadata type is performed (step 404). A search is performed by using the indices of the branch table. The indices of the
15 chosen metadata type are used and the condition of that metadata type is the key used in the search. Hence, this part of the search is done similar to any relational database. The rest of the work to perform a row search is filtering of the first set of rows (steps 405-413). Only
20 the rows that comply with the all the other conditions of the other metadata types are kept. Given the example branch table in Table 5 and the row searches b), d) and e) in Table 7, the resulting rows of those row searches are the rows in Table 9 below.

25

b)

0 [Adam]	0 [wife]	-	0 [person]	0 [person id]	0 [500101-2221]
----------	----------	---	------------	---------------	-----------------

d)

0 [Adam]	-	-	1 [person]	1 [person id]	1 [480101-1111]
----------	---	---	------------	---------------	-----------------

30

e)

0 [Adam]	-	-	1 [person]	2 [name]	2 [Adam]
----------	---	---	------------	----------	----------

Table 9. Resulting rows for each row search as given in Table 7.

35 Another possible way to perform a row search is of course to perform the searches for all metadata types and join the results regarding row number, but that will be less effective in most cases.

When all row searches have been done, the resulting rows are joined successively until the top level of the search hierarchy is reached (step 104). This process is described in detail in FIG. 5. The joins in the search hierarchy are done bottom-up (step 501) and may include several search units that have already been done and resulted in a number of rows. For all those search units (step 502) the algorithm gets the unique branch identifiers of the metadata type of the current join, by looking in the rows at the branch identifier, of the metadata type of the current join (steps 503-504). Given the example of join c) in Table 7 which joins at metadata type a, and the resulting rows from row search d) and e) in Table 9, the unique branch identifiers of metadata type a, are {1} for the rows from row search d) and {1} for the rows from row search e). All the sets of unique branch identifiers from the different search units are joined, and only the unique branch identifiers that have been found in all of the search units are kept (step 505). In the example join, the common unique branch identifiers from both row search d) and e) are {1}. For all kept branch identifiers (step 506) any row from any of the joined search units with that branch identifier is kept as the result of the join (step 507). Only one row for each branch identifier, from the search units, needs to be kept since the search units all have the same branch identifiers for all levels above the level at which they were joined. In the example of join c) the resulting rows are shown in Table 10:

30

c)

0 [Adam]	-	-	1 [person]	2 [name]	2 [Adam]
----------	---	---	------------	----------	----------

Table 10. Resulting rows after join c) as given in Table 8 of the rows from d) and e) in Table 9.

Using the same method the rows from row search b) and the rows from join c) are joined by join a) and results in the rows in Table 11:

35

0 [Adam]	0 [wife]	-	0 [person]	0 [person id]	0 [500101-2221]
----------	----------	---	------------	---------------	-----------------

Table 11. Resulting rows after join a) as given in Table 7 of the rows from b) in Table 9 and the rows from c) in Table 10.

5 In some cases it might be profitable to keep all the rows from all search units, but the resulting objects that are found will be the same.

The final step of the search is to calculate the resulting objects of the search by getting all unique
10 branch identifiers of metadata type o from the resulting rows. The resulting rows in Table 11 have only branch identifier 0 for metadata type o, and the result of that search is the object with branch identifier 0.

15 Metadata attributes

To each branch a number of metadata attributes can be attached. There can be restrictions as regards what attribute types are allowed to different metadata types. A more complete context free grammar to describe the
20 object structure is described in Table 12. Table 13 shows an example of an object with attributes attached to some of the branches. A branch table for the object is shown in Table 14.

Object	→	<o OAttributes> Relations </o>
OAttributes	→	Classification Begin End Datestatus Timestamp Owner Accessrights
Relations	→	Keys <r RAttributes> String Keys </r> Relations
RAttributes	→	Classification Reversename Begin End Datestatus Timestamp Owner Accessrights
Keys	→	Attributes <k KAttributes> String Attributes </k> Keys
KAttributes	→	Begin End Datestatus Timestamp Owner Accessrights
Attributes	→	Types <a AAttributes> String Types Attributes
AAttributes	→	Begin End Datestatus Timestamp Owner Accessrights
Types	→	Values <t TAttributes> String Values </t> Types
TAttributes	→	s
Values	→	s <v VAttributes> String </v> Values
VAttributes	→	Index Formattype Format Unit
Classification	→	c="String"
Reversename	→	r="String"
Begin	→	b="String"
End	→	e="String"
Datestatus	→	s="String"
Timestamp	→	m="String"
Owner	→	o="String"
Accessrights	→	a="String"
Index	→	i="String"
Formattype	→	t="String"

Format	→	f="String"
Unit	→	u="String"
String	→	s Character String
Character	→	\u0001 ... \uFFFF

Table 12. Context free grammar of the object structure

5	<o classification="person">Adam Anderson
	<a>person
	<t>firstname
	<v t="text">Adam</v>
	</t>
	<t>lastname
	<v t="text">Anderson</v>
10	</t>
	
	</o>
	<o classification="person">Adam Simpson
15	<a>person
	<t> firstname
	<v t="text">Adam</v>
	</t>
	<t>lastname
	<v e="2001-12-31" t="text">Anderson</v>
20	<v b="2002-01-01" t="text">Simpson</v>
	</t>
	
	</o>

Table 13. Simple example of objects with attributes.

25	O	r	k	a	t	v
	0 [Adam Anderson]	-	-	0 [person]	0 [firstname]	0 [Adam]
	0 [Adam Anderson]	-	-	0 [person]	1 [lastname]	1 [Anderson]
	1 [Adam Simpson]	-	-	1 [person]	2 [firstname]	2 [Adam]
	1 [Adam Simpson]	-	-	1 [person]	3 [lastname]	3 [Anderson]
	1 [Adam Simpson]	-	-	1 [person]	3 [lastname]	4 [Simpson]

Table 14. The branch table created from the data in Table 13.

The attributes to each branch can be stored and
 fetched from either primary memory or secondary memory or
 being stored in the inverted lists in the indices, all
 depending on whether higher speed or minimal use of
 primary memory is preferred. If the attributes are values
 where exact matches are of interest, they can be indexed
 just as metadata values.

Attribute constraints in queries

Queries can have constraint conditions on these attributes. An example of such a query described in xml is seen in Table 15:

```

5  <query end>="2002-01-01">
      <a>person
            <t>Anderson </t>
            <v t="text">Adam</v>
10  </a>
    </query>

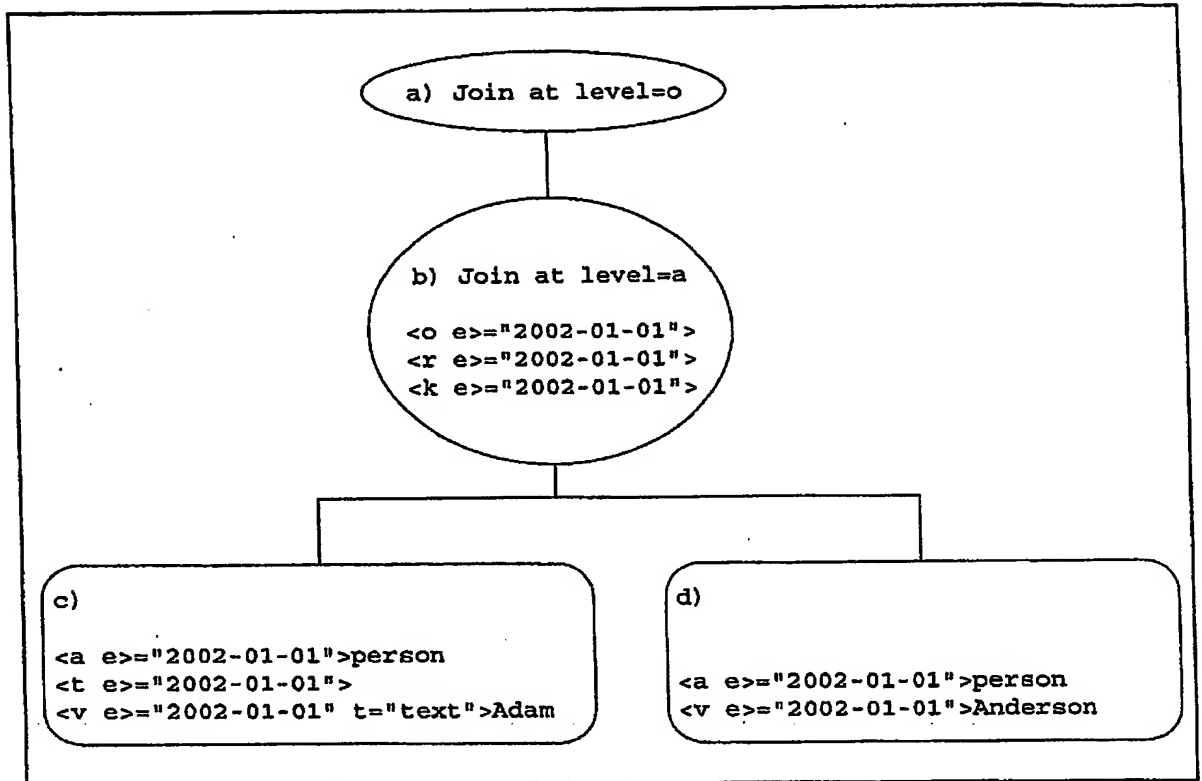
```

Table 15. Example of query in xml format with search criteria on attributes.

Note that `e>="2002-01-01"` is not correct xml, but it is used for reasons of simplicity. The constraints on attributes in queries do not always have to be equality constraints. Different types of attributes can allow different types of constraint conditions. Attribute constraints can be both general for the query, such as `e>="2002-01-01"` in the example query, or specific for a certain metadata value, such as `t="text"` in the example query.

Searching with attribute constraints

When using attribute constraints in the queries the search units created from the query need to have information on what attribute constraints there are on all the metadata types. Thus, the previous algorithm has to be amended (see Fig. 3). The structure of the search hierarchy and the conditions on metadata values are the same. To the row searches are added all attribute constraints, up to the metadata type of the preceding join (step 305). To the joins are added all attribute constraints, from the metadata type above the metadata type of the join, up to the metadata type of the preceding join (step 308). The example query in Table 14 results in the search hierarchy in Table 16:



5 Table 16. Resulting search hierarchy from the query described in Table 15

Even if there are no value conditions for a metadata type, there are cases when attribute constraints for that metadata type has to be considered. This depends on the different types of attributes and is not described in
10 further detail here.

The row searches are performed as before. Since checking rows against attribute constraints usually is more time consuming than joining sets of rows, it is
15 postponed until after the joins. The algorithm to perform successive joins until the top level is amended (see Fig. 6). In the amended algorithm all the resulting rows are filtered regarding attribute constraints (steps 606-610). As before the unique branch identifiers that are common
20 for all the search units that are joined are found (steps 602-605). For all the kept branch identifiers (step 606), the algorithm finds for each search unit that are joined, the rows with that branch identifier (steps 607-608).

Only the rows that comply with all of the attribute constraints of the search unit are kept (step 609). Only the unique branch identifiers that, after the attribute constraint filtering of rows, still has at least one row with that branch identifier in each of the search units that are joined are kept (step 610). The end of the join algorithm is as before. Only one row for each kept branch identifier is added to the result of the join (steps 611-612).

Given the example objects in Table 13 and its storage in the branch table in Table 14, a search with the query example in Table 15 gives the following result: The search hierarchy in Table 16 is created. All the row searches are performed and results in the rows in Table 17:

0 [Adam Anderson]	-	-	0 [person]	0 [firstname]	0 [Adam]
1 [Adam Simpson]	-	-	1 [person]	2 [firstname]	2 [Adam]

d)

0 [Adam Anderson]	-	-	0 [person]	1 [lastname]	1 [Anderson]
1 [Adam Simpson]	-	-	1 [person]	3 [lastname]	3 [Anderson]

Table 17. Resulting rows for each row search as given in Table 16.

20

When join b) is performed, the common unique branch identifiers of metadata type a, for the rows from c) and d), are {0,1}. The algorithm revisits the rows from c) and d) with those branch identifiers and removes the rows that do not comply with the attribute constraints in c) and d). The remaining rows are found in Table 18:

c)

0 [Adam Anderson]	-	-	0 [person]	0 [firstname]	0 [Adam]
1 [Adam Simpson]	-	-	1 [person]	2 [firstname]	2 [Adam]

30

d)

0 [Adam Anderson]	-	-	0 [person]	1 [lastname]	1 [Anderson]
-------------------	---	---	------------	--------------	--------------

Table 18. Remaining rows from c) and d) in Table 17 after filtering regarding attribute constraints.

The row from row search d) with the branch "

0 [Adam Anderson]	-	-	0 [person]	1 [lastname]	1 [Anderson]
-------------------	---	---	------------	--------------	--------------

Table 19. The resulting rows from join b) as given in Table 16 of the rows from c) and d) .

The join algorithm can be improved further if all the rows for each branch identifier are kept and the filtering regarding attribute constraints is postponed until the very last step when all joins have been made. The difference is that more information needs to be attached to each row that is kept, so when all joins have been made and the rows are being revisited, the algorithm knows what attribute constraints to apply to the rows and how to redo some of the joining. However the main idea is that filtering regarding attribute constraints is made after the joining of rows in the algorithm. This improves the speed of the searches since a join is fast while attribute values may not always be quick to access from memory and the number of rows after the join of rows is usually much less than the initial number of rows.

Relations

The structure of objects with the specified metadata types (o,r,k,a,t,v) makes it simple to define relations between objects. The branches underneath an r-branch can be the metadata to search for in other objects. A relation may not have any match in any other object and

may have more than one match. A relation from an object is not a fixed "pointer", but a value reference or a search for other objects. Consider the two objects "Adam" and "Eve" in Table 20 and Table 21:

5

	<code><o>Eve</code>
	<code><a>person</code>
	<code><t>person id</code>
10	<code><v>000108-2221</code>
	<code></v></code>
	<code></t></code>
	<code></code>
	<code></o></code>

Table 20. The object Eve

15

	<code><o>Adam</code>
	<code><a>person</code>
	<code><t>person id</code>
20	<code><v>000106-1111</code>
	<code></v></code>
	<code></t></code>
	<code></code>
	<code><r>wife</code>
25	<code><a>person</code>
	<code><t>person id</code>
	<code><v>000108-2221</code>
	<code></v></code>
	<code></t></code>
	<code></code>
30	<code></r></code>
	<code></o></code>

Table 21. The object Adam with a relation "wife" to the object Eve in Table 20.

35 The object Adam has a relation wife to Eve. Under the r-branch "wife", Adam has metadata that identifies the object Eve. The algorithm to find other objects that a particular object has relations to is given in FIG. 10 (steps 1001, 1002-1005). Given an assumed object (step 1001), all its r-branches are transformed to queries (step 1003). The query has the same metadata as the relation branch, but the top condition is changed to `<r>-`, which determines that no value is allowed at that level. This condition prevents the query to find the assumed object and all other objects with similar relations. The query is performed (step 1004) and the result of the

40

45

search is the objects that the relation points at (step 1005). The relation "wife" in Table 21 is transformed into the query in Table 22:

```

5  <query>
    <r>-
      <a>person
        <t>person id
          <v>000108-2221
10         </v>
          </t>
        </a>
      </r>
    </query>

```

15 Table 22. The query created from the relation "wife" in Table 21.

The result of this query is the object Eve.

Reverse relations

20 It is useful to be able to know what other objects refer to a known object. This is possible in the present invention by using the metadata type k (key). A key declares some attributes in an object as unique or important identifiers of the object. In the object
 25 example Eve in Table 23 a key "person id key" declares the person id as an identifier of the object.

```

30 <o>Eve
    <k>person id key
      <a>person
        <t>person id
          <v>000108-2221
          </v>
        </t>
      </a>
    </k>
35 </o>

```

Table 23. The object Eve with a defined key "person id key"

40 Finding reverse relations is similar to finding relations. The algorithm to find other objects that a particular object has reverse relations to is given in FIG. 10 (steps 1001, 1006-1009). Given an assumed object (step 1001), all its k-branches are transformed to

queries (step 1007). The query has the same metadata as the key branch, but the top condition is changed to `<r>*`, which determines that a value is required at that level. Only objects with a relation to those values are found.

- 5 The query is performed (step 1008) and the result of the search is the objects that have a relation that points at the assumed object (step 1009).

These queries will only find objects that have relations to the values in the keys, but since keys are
10 chosen to mark unique or important attributes in objects, it is very likely that most relations refer to the values in keys. To find the reverse relations from the object Eve in Table 23, the key "person id key" is transformed into the query in Table 24:

15

```

20 <query>
    <r>*
        <a>person
        <t>person id
            <v>000108-2221
            </v>
        </t>
    </a>
    </r>
25 </query>

```

Table 24. The query created from the key "person id key" in Table 23.

The example query finds the object Adam in Table 21 but not the object Eve in Table 23.

30

Associations

In the present invention it possible to define associations between objects. An association is a directed connection between two objects. An association
35 map is a set of such associations. The association maps are used for several purposes: 1) to allow inheritance of metadata from objects. Metadata is inherited in the direction of the associations. An object inherits all metadata from the objects that has an association to it
40 and also all metadata from objects that are indirectly

associated to the object through several associations. This allows new ways to organize objects. For example, if a search or relation finds an object, all objects that inherit from that object are also found. 2) To store away
5 relations between objects to give a faster way to browse between objects. Relations between objects are found by performing searches, which might sometimes not be fast enough. When a relation once has been found, an association between the same two objects can be created. This
10 allows all relations to be pre-calculated and when a user wishes to browse between objects, it can be done without performing any time consuming searches. 3) To organize objects by a model. It is possible in the present invention to define models describing how different types
15 of objects are related to each other. When a new object is inserted into the present invention, all relations to and from that object are found. For the relations that match a definition in a model, a corresponding association is created and the new object has automatically
20 been organized.

Searching with inheritance

When performing a search any set of association maps can be used for inheritance. The same search query can
25 give different results depending on which association map that is used.

When using inheritance of metadata the search algorithm in FIG. 1 is slightly changed. The new algorithm is described in FIG. 7. The top-level join is not performed
30 along with the other joins in the search hierarchy (step 704). All the search units that are joined at the top level are handled separately (step 705). First the rows are filtered regarding attribute constraints (step 706) and the resulting objects are calculated (step 707). Then
35 from the resulting set of objects, all inheriting objects are added to the resulting set of objects for that search unit (step 708). Inheriting objects are found by

following all associations, in the set of association maps used for inheritance in the current search, from the current set of objects. The resulting sets of objects are then finally joined and the end result of the search is found (step 709).

Searching with inheritance may cause performance problems, since filtering regarding attribute constraints sometimes has to be done on large sets of rows even if the end result of objects is not very large, and should therefore be used with caution. But the possibility to find complex relations such as "which objects do both Adam and Eve have a relation to?" may sometimes be worth waiting for.

15 **Searching simplified**

The storing and search procedure of the present invention have several great advantages over traditional searches in databases: 1) No initial knowledge of the structure of the data in the database is needed. All values for each metadata type can be searched for. 2) Queries with complex structures can be made. 3) Queries are simply matches with metadata in the database. No complex query language needs to be handled. 4) Searching is similar to free text searching on Internet. All values, regardless of where in the database it is, can be searched for in the same query. In a traditional database, the column where the search is performed has to be specified. 5) The search result is not just a number of rows in a database table. It is one or more complete objects that besides the data proper, also contains relations to other objects.

This way of putting all data in the whole database into one single table is possible since a few allowed metadata types in a specified order (o,r,k,a,t,v) has been chosen, and all data has to be described in this format. Since the number of metadata types is known, a table can be created with the same columns in the same

order as the metadata types. The complex tree like structure of an object is retained in the branch table by repeating branches for each time its children occurs in a row. With these methods internal data storage, indexing
 5 and searching are made possible.

Why the proposed generalized search gives good performance

In a traditional relational database searching is
 10 quicker since not all values lies in the same column. The drawback in performance of the present invention is only that the branch table is bigger than the tables in a traditional database.

In the present invention, all the columns in the
 15 branch table are indexed. Since each row search choose to perform searches in the indices only for the condition that will give the least number of rows, and the rest of the conditions are checked on that set of rows, the minimum number of rows to fetch from indices are chosen.

20 Joins of search units are no different in complexity than joins between tables in a traditional database. A join always has to match all rows from one set of rows to all rows in another set of rows. Note that in the present invention a user does not have to specify how this join
 25 is going to be made.

Implementation

A practical implementation of the current invention has been done by programming in the object-oriented
 30 programming languages Java and C#. A modular illustration of the realization of the current invention is given in FIG 8. In FIG.8, a CapishBaseEngine module 800 serves as an entry point for user logon, search queries, data modification, etc. A CapishBaseDataStorage module 802
 35 handles all queries, data modification, etc., as well as translating these to the appropriate internal format. A DataStorageCore module 804 serves to handle queries, data

modification, etc., on a lower level and also to make sure that changes will be consistent throughout the system. It also controls an Associations module 808 which manages the associations in the system.

5 Indexing and searching is performed by a DataStorageSearchIndex module 806. To this end, it cooperates with a BranchTable module 810 and a ColumnIndices module 812, where the former handles the branch table and the latter contains the indices to the
10 columns of the branch table.

 Both the DataStorageCore module 804 and the DataStorageSearchIndex module 806 cooperate with a Metadata module 814, which contains all metadata and values, and makes them accessible and searchable. To this
15 end, the Metadata module 814 uses an AttributeValues module 818 and a MetadataValues module 820, which contain all attribute values and metadata values, respectively, that belong to the branches stored in the branch table.

20 Distributed Network Architecture

 Answers to questions often require accessing information from multiple independent data sources. The current invention allows for integration of data from different sources over any distributed network, since all
25 related objects are found by executing specific relation queries over the network. One possible network solution for the current invention is shown in FIG. 9. A client 930 is connected to a router 920 over a wide area network 900 such as the Internet. The router 920 is connected to
30 any number of other routers 922, as indicated at 921 and 923. The routers 920, 922 may be connected to different servers 902, 904, 906, 908. Each server has any number of local databases 916, 918. A user on the client 930 can
35 from an existing network choose any number of local databases to work with. Each query or other operation submitted by the client 930 to the router 920 is distributed through the network 900 to the chosen set of

local databases 916, 918 via their associated server 906. The answers from the local databases 916, 918 are returned and joined to a common result in the servers and the routers, and ultimately presented to the user on the client 930.

The invention has mainly been described above with reference to a few embodiments. However, as is readily appreciated by a person skilled in the art, other embodiments than the ones disclosed above are equally possible within the scope of the invention, as defined by the appended patent claims.

CLAIMS

1. A computer program product having program code adapted to provide, upon execution, a database of objects and a database engine for managing said objects,
5 characterized in that

each object comprises metadata for describing the object and for defining a hierarchical structure of branches which constitutes said object and which includes
10 relations to other objects,

wherein an individual branch has:

a metadata type which is selected from a predetermined and limited set of n different metadata types;

15 a metadata value; and

an arbitrary number of other branches connected thereto, as children of said individual branch, said arbitrary number including zero branches, and

wherein each metadata type represents one respective
20 level in said hierarchical structure.

2. A computer program product as in claim 1, wherein each object is stored in said database in the form of an n -field data structure for each particular branch that
25 does not have any children, and wherein each field of the n -field data structure represents a respective branch that preceds said particular branch as its ancestor.

3. A computer program product as in claim 1 or 2,
30 wherein said database comprises a branch table having columns that correspond to the n different metadata types and wherein said n -field data structure is stored as a row in said branch table.

35 4. A computer program product as in any preceding claim, wherein said database engine has a query function for searching said database and wherein said query

function is adapted to accept search queries which are specified in a declarative language in accordance with said hierarchical structure of objects in said database.

5 5. A computer program product as in claim 4, wherein said declarative language is xml.

6. A computer program product as in any preceding claim, wherein $n = 6$ and said set of metadata types
10 consists of {Object, Relation, Key, Attribute, Type, and Value}, metadata type Object representing the root level of said hierarchical structure.

7. A computer program product as in claim 3 and 4,
15 wherein said query function for searching said database comprises the steps of:

- a) receiving a search query;
- b) transforming the search query into a search hierarchy of row searches and joins by:
 - 20 b1) generating a top join for joining all results at metatype Object;
 - b2) traversing the branches of the search query top-down by
 - 25 b'1) if a particular branch has no children, creating a row search with the metadata values of all branches from the root down to the particular branch as conditions on metadata values in the row search;
 - 30 b'2) if a particular branch has more than one child, creating a join for joining row searches and joins at the metatype of the particular branch;
 - b'3) adding a row search created in step b'1) or a join created in step b'2),
35 respectively, as a child of a preceding join;
 - c) performing all the row searches from step b) so as to result in a set of rows in the branch table;

d) performing the joins in the search hierarchy successively starting from the bottom of the search hierarchy and ending with the top of the search hierarchy; and

5 e) producing a result of the search query by retrieving all unique objects from the outcome of step d).

8. A computer program product as in any preceding
10 claim, wherein said metadata includes metadata attributes to respective branches of an object.

9. A computer program product as in claim 4 and 8,
15 wherein said query function is adapted to accept search queries with constraint conditions on said metadata attributes.

10. A computer program product as in claim 9,
20 wherein said metadata attributes include timestamps on individual branches of an object.

11. A computer program product as in claim 9,
wherein said metadata attributes include access rights to individual branches of an object.

25

12. A computer program product as in claim 9,
wherein said metadata attributes include unit definitions for individual branches of an object.

30 13. A computer program product as in claim 4,
wherein a branch of a predetermined first metadata type in an individual object defines a relation from said individual object to another object and wherein said query function is adapted to search said database to find
35 said another object by matching the metadata thereof with the metadata included in said branch of a predetermined first metadata type value in said individual object.

14. A computer program product as in claim 13,
 wherein a branch of a predetermined second metadata type,
 other than said first metadata type, in an individual
 5 object allows another object to define a reverse relation
 to said individual object, in the form of a branch of
 said predetermined first metadata type in said another
 object, and wherein said query function is adapted to
 find said another object by matching the metadata
 10 included in said branch of a predetermined second
 metadata type in said individual object with the metadata
 included in said branch of a predetermined first metadata
 type in said another object.

15 15. A computer program product as in claim 6, 13 and
 14, where said first metadata type is Relation and said
 second metadata type is Key.

16. A method of searching a database of objects as
 20 provided by the computer program product according to any
 of claims 1-15, characterized by the steps of
 specifying, through a man-machine interface of a
 computer (930), a search query in a declarative language
 in accordance with said hierarchical structure of objects
 25 in said database;

submitting said search query to said database engine
 through said computer;

receiving a result of said search query at said
 computer; and

30 presenting said result through said man-machine
 interface.

17. A method of exporting data from at least one
 arbitrary database, characterized by
 35 identifying a first plurality of relations between
 different pieces of data in said at least one database;

defining a second plurality of objects, each of said objects comprising metadata which represents individual ones of said first plurality of relations and said different pieces of data, said metadata being of n different metadata types, n being a predetermined integer;

for each object, forming an n -level hierarchical information structure, where each metadata type is represented at a respective unique level; and
 10 storing the information structures thus formed for said second plurality of objects.

18. A method as in claim 17, where said information structures are stored in at least one flat file on a long-time data storage medium, including but limited to a magnetic disk, an optical disk, a magnetooptical disk or an electronic semiconductor memory.

19. A method as in claim 17 or 18, where said information structures are expressed in a text format of a declarative language.

20. A method as in claim 19, wherein said declarative language is xml.

25

21. A method of connecting objects in a database of objects as provided by the computer program product according to any of claims 1-15, characterized by the steps, for an assumed individual object, of

30 for all branches in the assumed object that are of a first predetermined metadata type, said first predetermined metadata type allowing a relation to be defined from the assumed object to another object:

35 forming a query based on the metadata of the branch,

searching the database with the query, and

10 forming a query based on the metadata of the
 branch,
 searching the database with the query, and
 collecting, as a result of the searching step,
 all objects, if any, that have a relation to the
 assumed object.

15

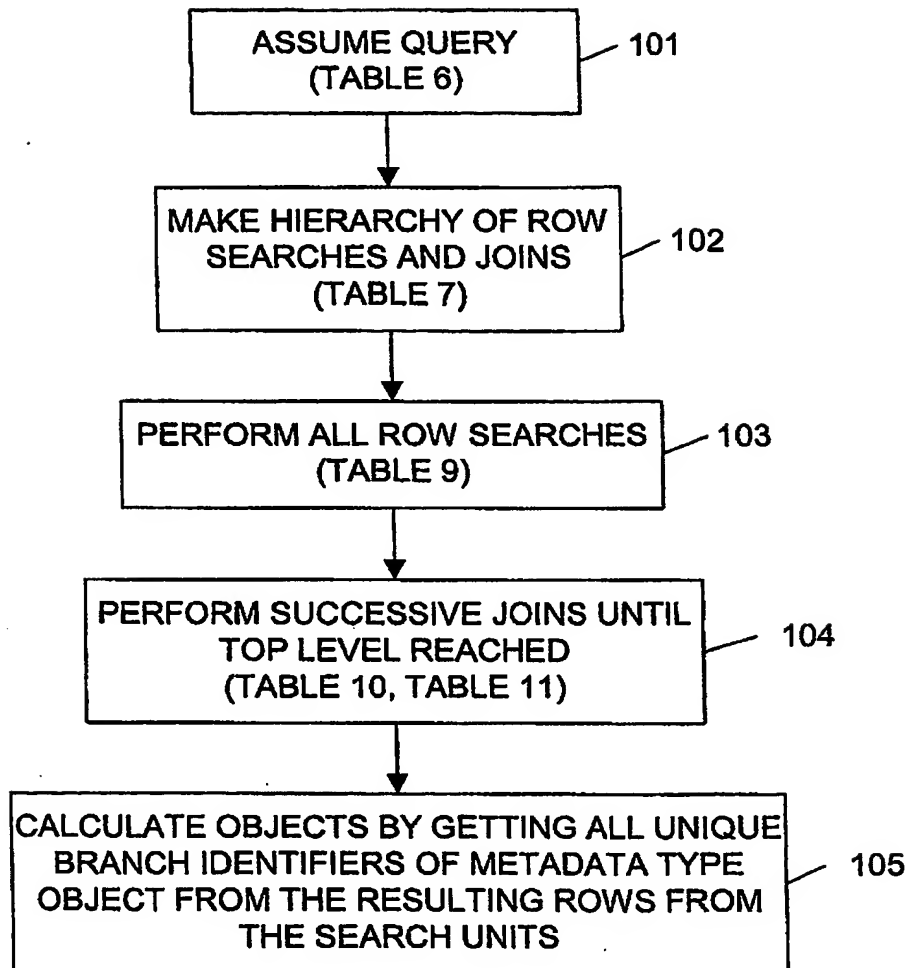
ABSTRACT

A computer program product has program code adapted to provide, upon execution, a database of objects and a
5 database engine for managing the objects. Each object comprises metadata for describing the object and for defining a hierarchical structure of branches which constitutes the object and which includes relations to other objects. An individual branch has a metadata type,
10 which is selected from a predetermined and limited set of n different metadata types, and a metadata value. An individual branch may also have an arbitrary number of other branches connected thereto, as children of the individual branch, wherein the arbitrary number includes
15 zero branches. Each metadata type represents one respective level in the hierarchical structure.

To be published with FIG 8.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185

1/10

*Fig 1*

2/10

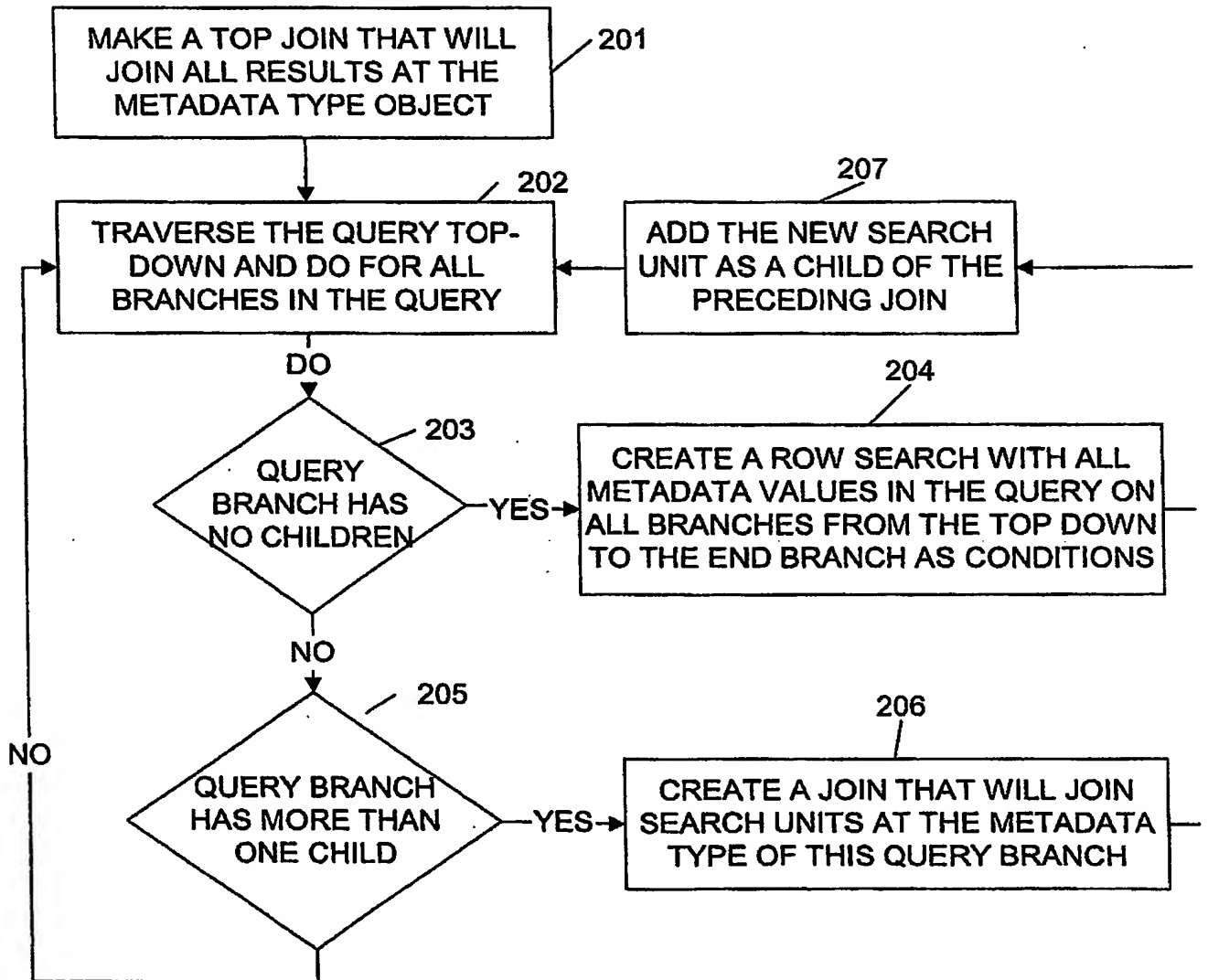


Fig 2

3/10

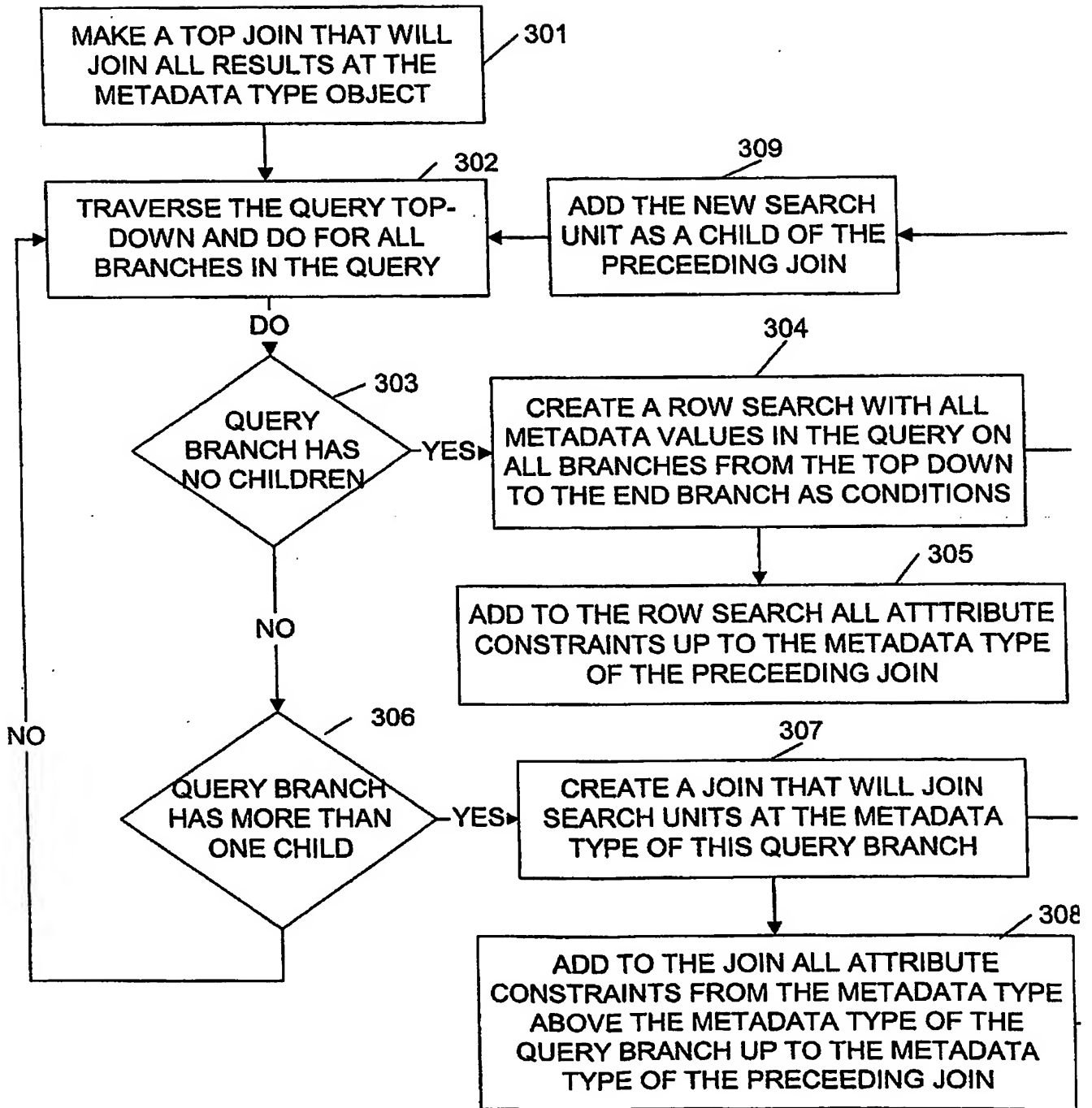


Fig 3

4/10

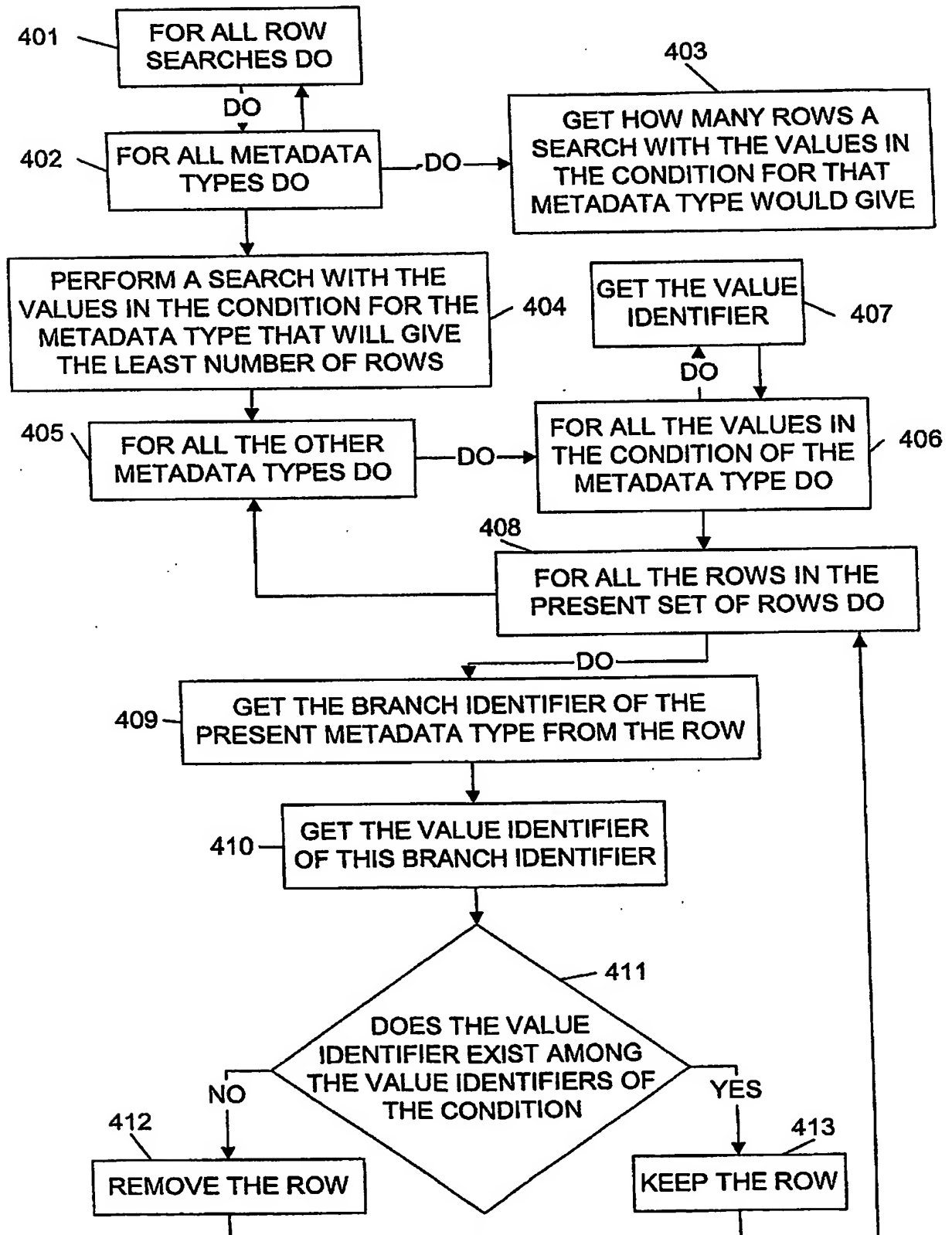


Fig 4

5/10

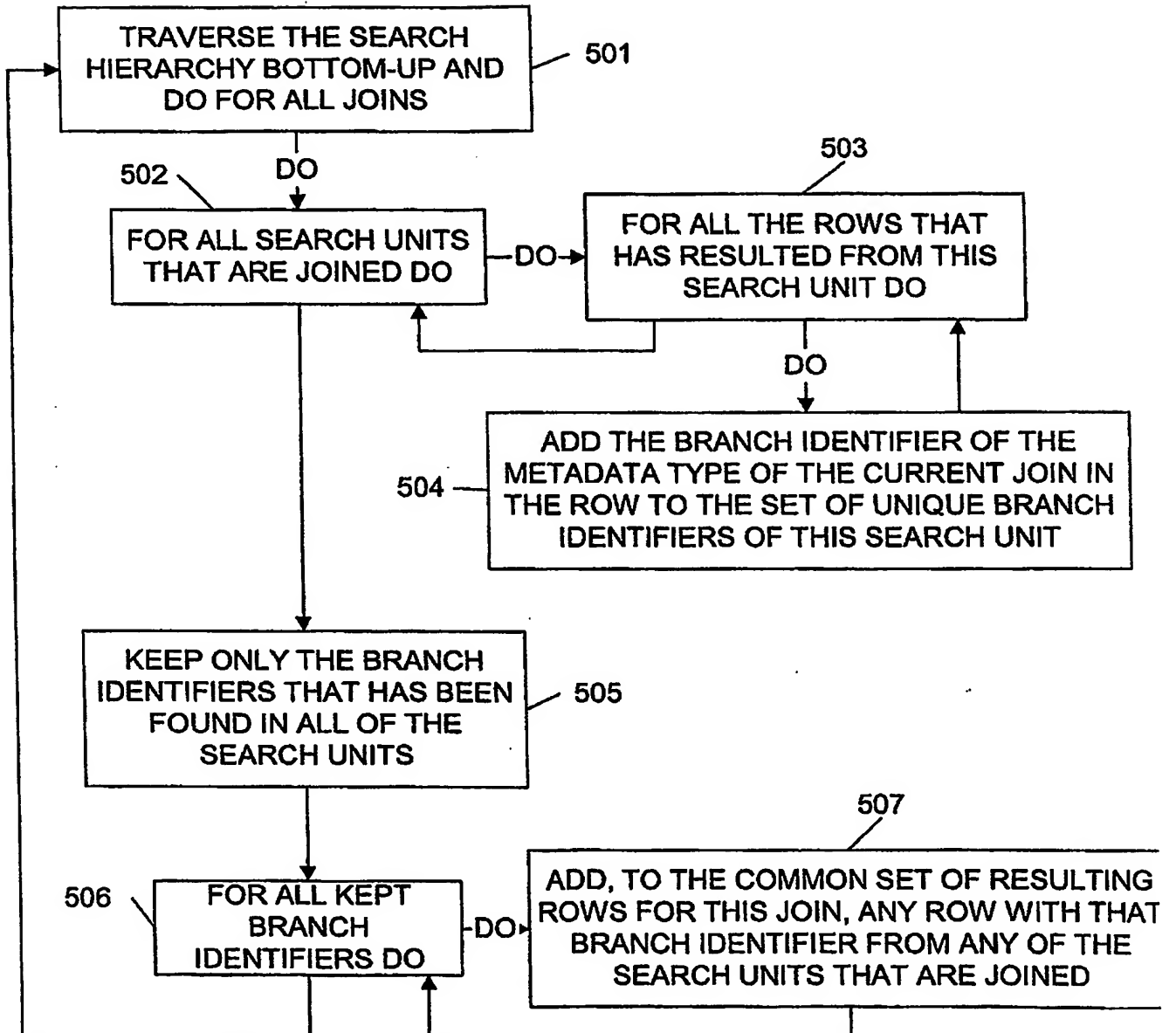


Fig 5

6/10

FIG. 6

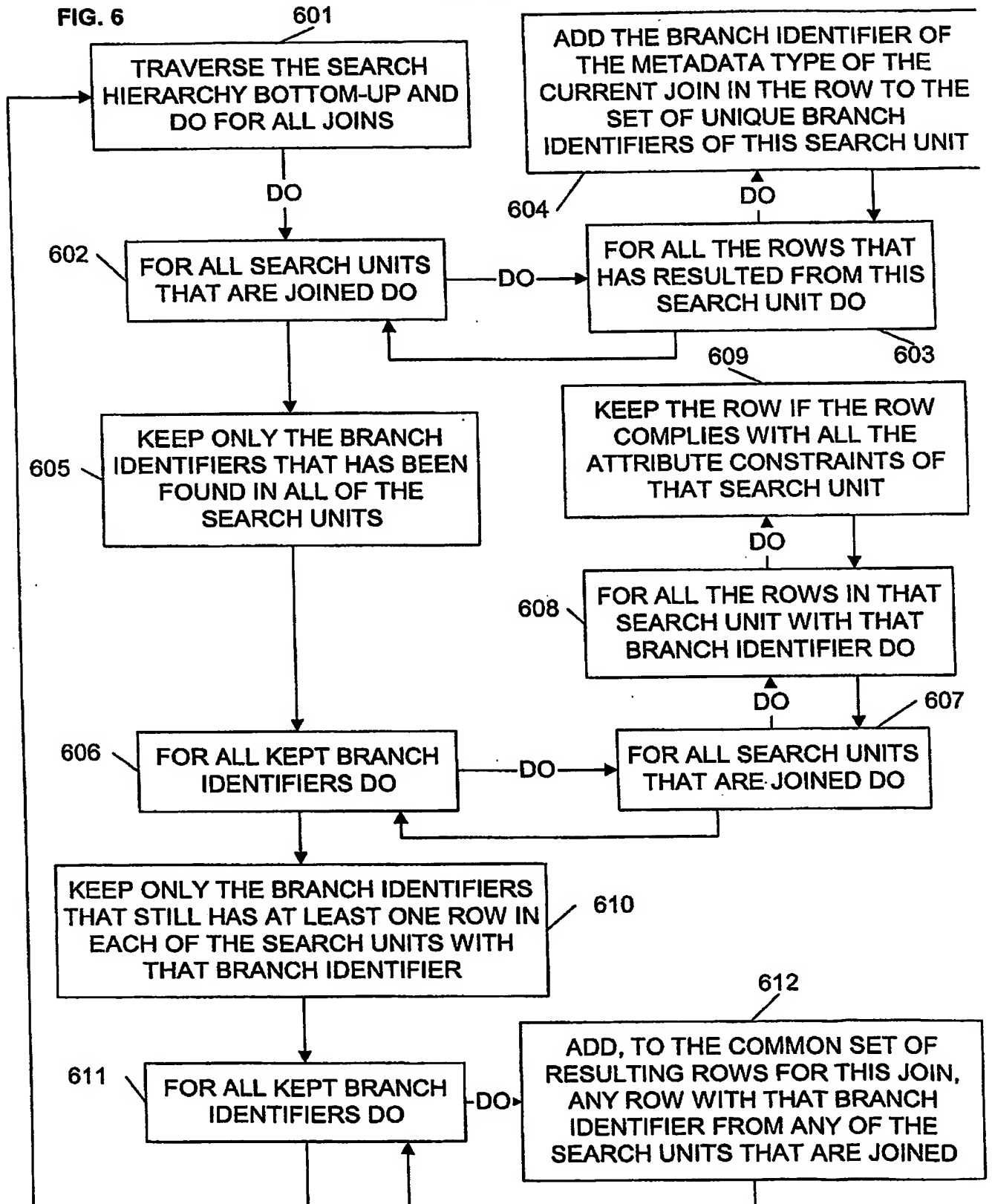


Fig 6

7/10

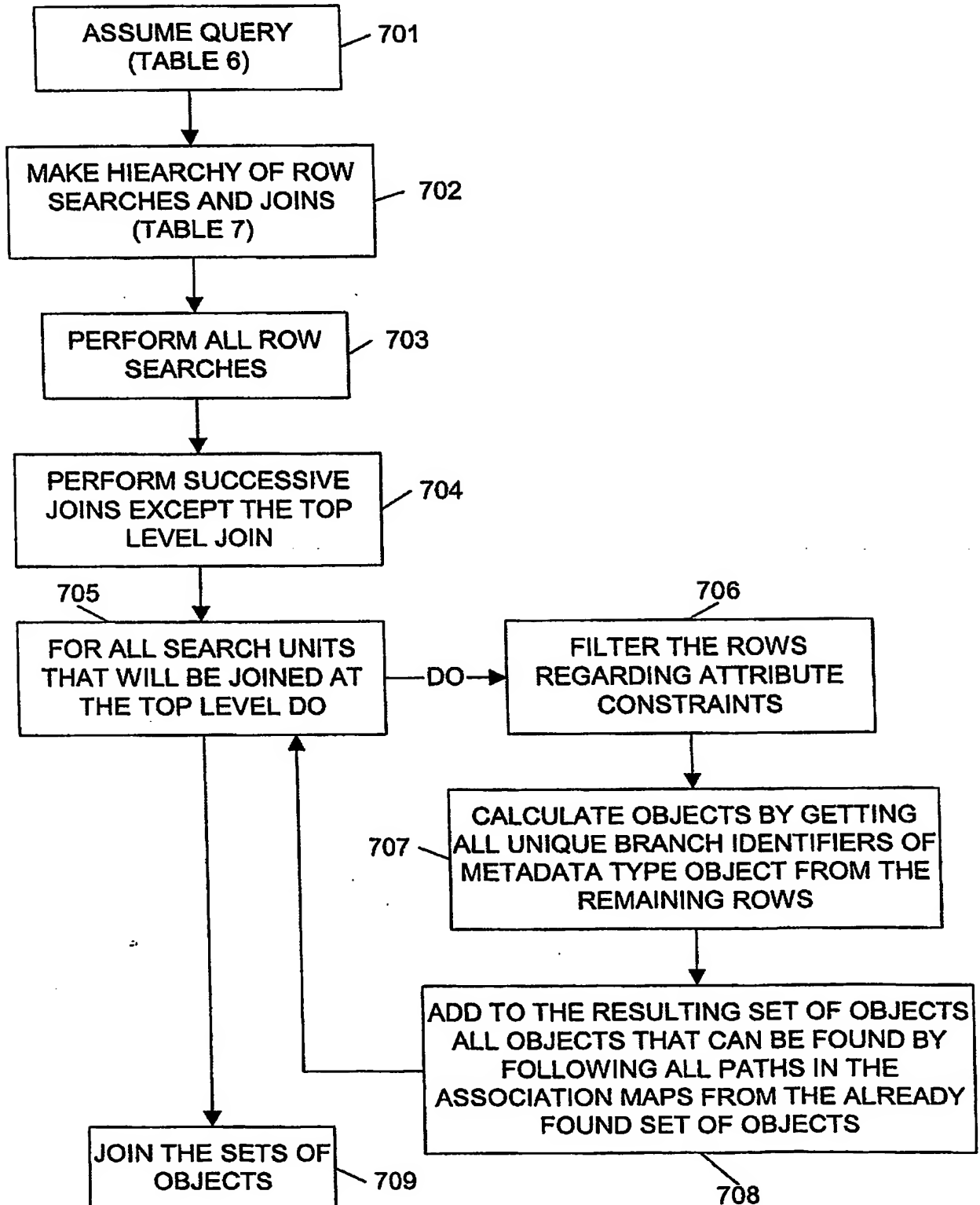


Fig 7

8/10

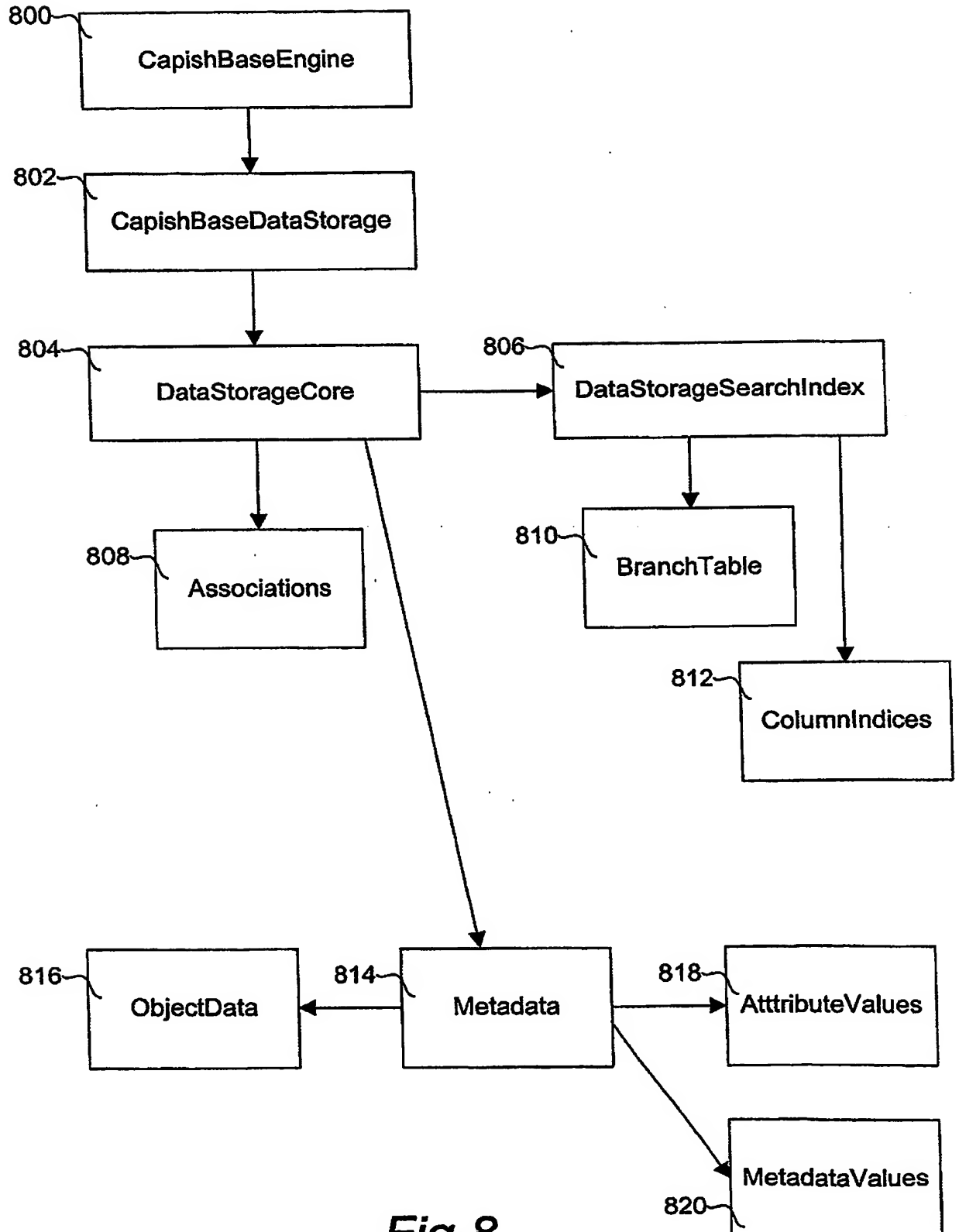
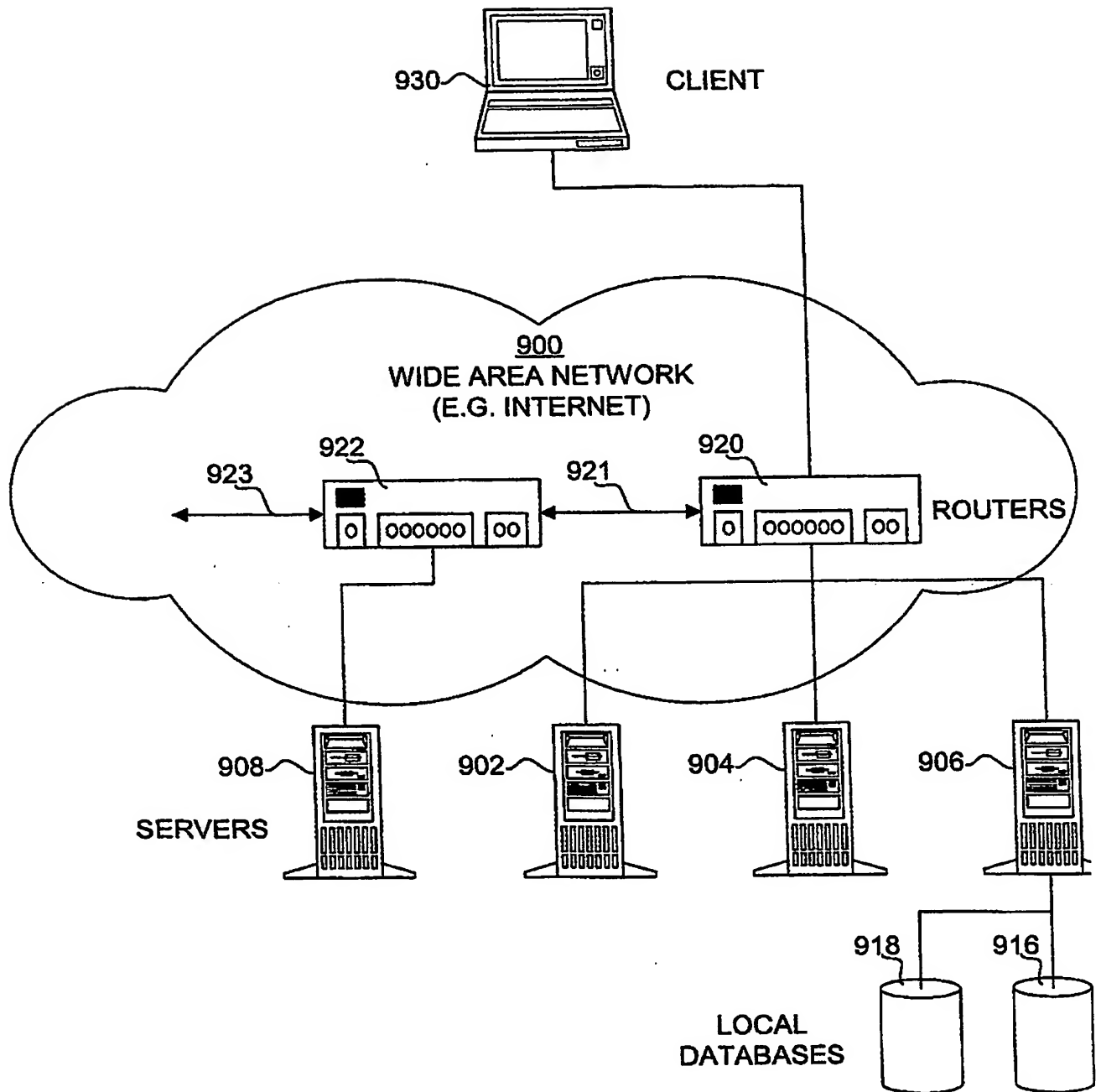
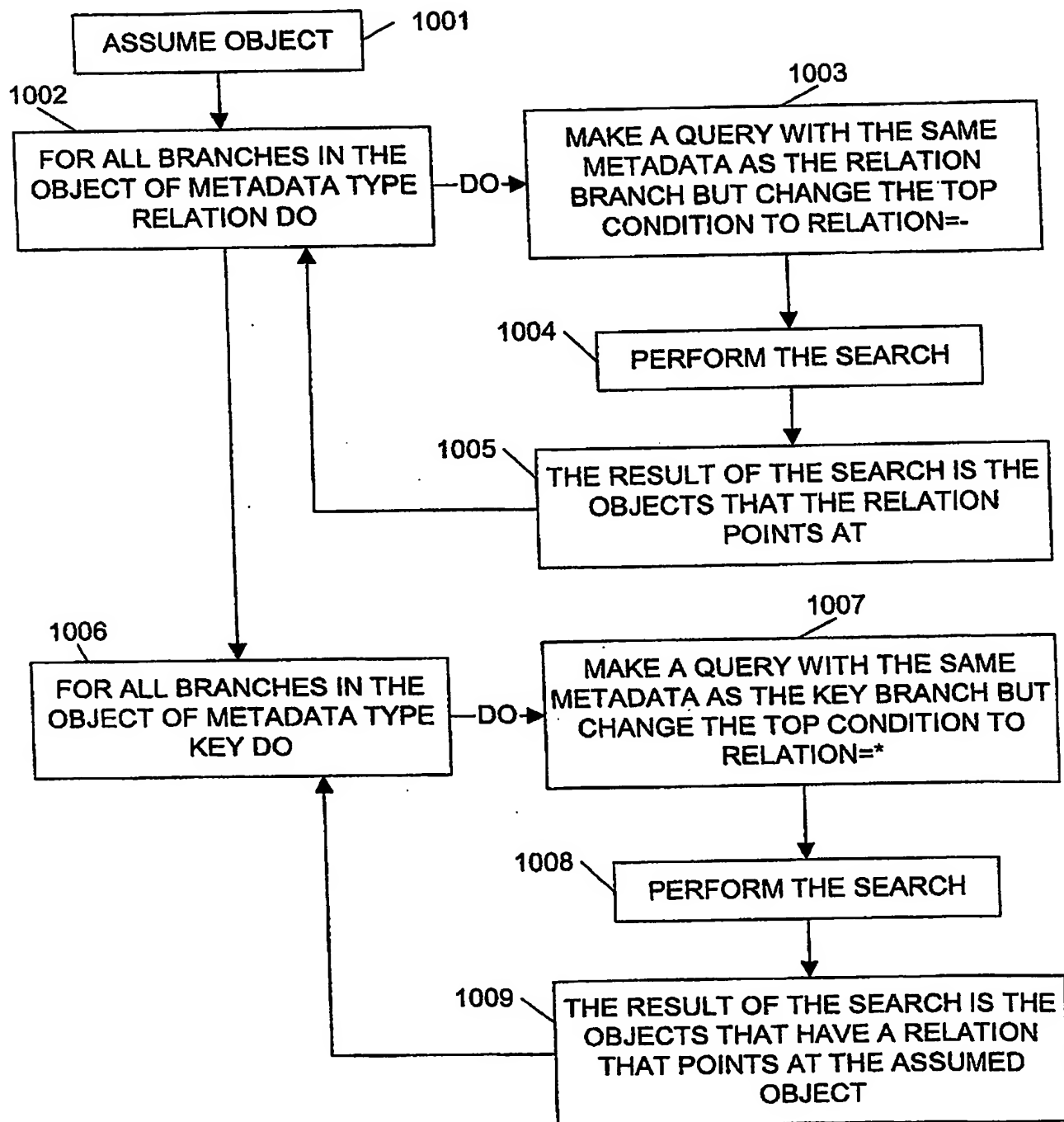


Fig 8



10/10



**This Page is Inserted by IFW Indexing and Scanning
Operations and is not part of the Official Record**

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☐ FADED TEXT OR DRAWING
- ☒ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☐ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: _____

IMAGES ARE BEST AVAILABLE COPY.

As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.